Attorney Reference: 109870-130097 *Patent*
IPG No: P009

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:	)	Examiner: Rampuria, Satish
Bosworth et al.	)	Art Unit: 2191
Application No.: 10/089,139	)	Customer No.: 25,943
Filed: August 19, 2002	)	
Conf. No.: 2275	)	
For: A MULTI-LANGUAGE EXECUTION METHOD	)	

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

REPLY TO EXAMINER'S ANSWER

Dear Sir:

Appellants respectfully reply to the Examiner's answer as follows:

(A) In "Response to Argument," page 14, the Examiner notes that the rejection of claims 20-32 under §112, second paragraph, was based on a typo in the heading and removes the §112 rejection of claims 20-32. In response, Appellants thank the Examiner for acknowledging the typo and removing the rejections.

(B) In "Response to Argument," pages 14-16, the Examiner responds to Appellants' argument that Wang fails to expressly or inherently disclose the recitations of claim 1. As stated by Appellants' Appeal Brief, claim 1 requires that a common execution engine recognize code sections of two programming languages and invoke code statement processing units for each section. Appellants noted that Wang fails to teach any such common execution engine because the Java VM,

which the Examiner equates to the first code statement processing unit, invokes the VB script interpreter, which the Examiner equates to the second code statement processing unit. Thus, rather than a common execution engine that invokes two processing units, Wang teaches a processing unit for a first language that periodically invokes a processing unit for a second language.

In response, on page 15, the Examiner first reiterates a point of non-contention: that Wang recognizes different input sources and different processors to process those sources. Then, the Examiner concedes the Appellants' above-summarized characterization of Wang. According to the Examiner: "Wang discloses during runtime Java Virtual machine and VisualBasic Script interpreter execute their respective corresponding sources. Java Virtual machine performs normal processing of intermediate source until it invokes a thread object run method to initiate (invoke) the execution of VisualBasic Script interpreter" (emphasis added). Thus, the Examiner recognizes that it is the first code statement processing unit (Java VM) that invokes the second unit (VB Script Interpreter) in Wang, not a common execution engine that recognizes sections of both languages and invokes both processing units, as recited by claim 1. Accordingly, the Examiner concedes Appellants' argument but fails to recognize its necessary conclusion.

Further on page 15, the Examiner reasserts that the term "common execution engine" is not present in either Appellants' claims or specification. In Appellants' Appeal Brief, Appellants addressed this previously-expressed concern of the Examiner's. The Examiner, however, chose not to respond to Appellants' remarks. Thus, those remarks are worth repeating in full:

> In the Advisory Action mailed May 14, 2007, the Examiner asserts that Applicants' argument relies on features which are not recited in the rejected claims. More specifically, the Examiner states that Applicants' argument relies on a "common process", and that no common process is recited in the claims. As can be seen above, Applicants' argument relies on a "common execution engine" that invokes the first and second code statement processing units. While "common" is not explicitly recited in the rejected claims, "execution

> engine" is, and that execution engine is inherently a "common execution engine." In interpreting the language of claim 1 above in light of the antecedent basis rule, the Examiner is required to treat "the execution engine", which is recited as invoking both the first and second processing units, as the same execution engine. This is what Applicants mean when Applicants argue that claim 1 teaches that the first and second code statement processing units are invoked by a common execution engine. Accordingly, Applicants' above arguments stand unrefuted.

Also, on page 15, the Examiner claims to accept, for the sake of argument, the common execution engine, and argues that Wang discloses a runtime processor which invokes both the Java VM and the VB Script Interpreter. This argument, however, directly contradicts the Examiner's above-quoted admission that the VB Script Interpreter is invoked by the Java VM. Further, the Examiner's above-quoted admission is well-supported by Wang in Col. 3, lines 50-67. The passage of Wang quoted for the Examiner's argument that Wang teaches a common execution engine, col. 2, lines 49-55, teaches no such thing. That passage merely cites that, in a preferred embodiment, the system of Wang includes two runtime processors 110 and 112, a Java VM to execute Java statements and a VB Script Interpreter to execute VB Script statements. Thus, the Examiner fails to point out anywhere that Wang expressly or inherently discloses a common execution engine that recognizes multiple languages and invokes processors for each language. In fact, in admitting that the first runtime processor (Java VM) invokes the second (VB Script Interpreter), the Examiner establishes that Wang teaches no such common execution engine. Accordingly, claim 1 is not anticipated by Wang under §102.

Lastly, as argued by Appellants in their Appeal Brief, this lack of a common execution engine isn't a minor difference that might be suggested to one of ordinary skill in view of Wang. Rather, it is an alternative method of processing multi-language specifications that actually teaches away from Appellants' execution engine. As Appellants stated in the Appeal Brief:

> Both Wang and the invention of claim 1 certainly teach methods of processing multi-language specifications. But Wang teaches an alternative solution to that proposed by the claimed invention of claim 1. In Wang, no common execution engine controls the "hand off" of execution between the code processing units. Thus, Wang inserts synchronizer tokens into the intermediate code to eliminate the need for a common execution engine. In contrast, the claimed invention of claim 1 does teach such a common "execution engine" that controls the invoking of both code processing units. Thus, in the claimed invention of claim 1, no synchronizer tokens are needed. Accordingly, for the reasons given above, Wang does not anticipate claim 1 and, because Wang proposes an alternative solution teaching away from that of claim 1, Wang does not even suggest claim 1.

Therefore, claim 1 is patentable over Wang under both §102 and §103.

(C) In "Response to Argument," pages 16-17, the Examiner further responds to Appellants' argument that Wang fails to expressly or inherently disclose the recitations of claim 6. As stated by Appellants' Appeal Brief: "Claim 6 recites recognizing a third code section of a third language and invoking a third code statement processing unit of the third language to process the third section." Wang, according to the Appellants, only teaches two runtime processors to process two intermediate sources of two languages.

In response, the Examiner states that Wang is not limited to two programming languages, and cites col. 6, lines 21-25 of Wang as teaching "that the invention is not limited to specific programming languages, and could comprise languages other than HTML, Java, VisualBasic Script, [such as,] for example, C, C++, Perl, Cobol etc."

Appellants acknowledge that Wang does indeed teach that other languages may be used. But in teaching that other languages may be used, Wang in no way expressly or inherently requires a third code section of a third language and a third code statement processing unit. Rather, Appellants respectfully submit that col. 6, lines 21-25 of Wang simply teaches that other languages may be used in place of Java and VB Script. To anticipate claim 6, Wang would need to expressly recite a

code section of a third language and a third runtime processor for the third language or at least inherently disclose such elements. But Wang does not expressly recite such elements, and the figures and passages cited by the Examiner do not inherently require them. Accordingly, claim 6 is not anticipated by Wang under §102.

Also, Appellants respectfully note that adding a third processor and a third language would be no simple modification of Wang. As discussed above, rather than having a common execution engine capable of recognizing different languages, Wang relies on the use of synchronization tokens to cause a processor of a first language to invoke a processor of a second language. Adding a third language would require the processor of the first language to differentiate between tokens of two different languages rather than simply recognizing a token. This could require significant reprogramming of existing tokens and of the processor capable of recognizing them. Thus, Appellants respectfully submit that a person of ordinary skill in the art would not find any suggestion to modify Wang to include a third runtime processor for handling a third language. Accordingly, Appellants submit that claim 6 is patentable over Wang under both §102 and §103.

Conclusion

As Appellant has set forth in the brief, the Examiner has erred in his rejections. Accordingly, Appellant respectfully requests that the Board reverse the Examiner's rejections.

Please charge any shortages and credit any overages to Deposit Account No. 500393.

Respectfully submitted,
Schwabe, Williamson & Wyatt, P.C.

Date: March 20, 2008

/Robert C. Peck/
Robert C. Peck, Reg. No. 56,826
Agent for Appellant

Pacwest Center
1211 SW Fifth Ave., Ste 1600-1900
Portland, Oregon 97204
Phone: (503) 222-9981,
FAX: (503) 796-2900